

ShadowICS: Detecting OT Sandboxes via Passive Industrial Traffic Monitoring

Denis Donadel¹[0000–0002–7050–9369], Daniele Antonioli²[0000–0002–9342–3920],
Nils Ole Tippenhauer³[0000–0001–8424–2602], and Massimo
Merro¹[0000–0002–1712–7492]

¹ University of Verona, Verona, Italy
{denis.donadel,massimo.merro}@univr.it
² EURECOM, Sophia Antipolis, France
daniele.antonioli@eurecom.fr

³ CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
tippenhauer@cispa.de

Abstract. Industrial Control Systems (ICSs) are increasingly targeted by sophisticated OT malware, yet the dynamic analysis tools used to study such threats rely on virtualized environments whose realism remains largely unverified. Advanced IT malware routinely detects sandbox artifacts to evade analysis, but whether analogous evasion strategies are feasible in OT settings, where industrial traffic and physical-process dynamics offer distinctive fingerprints, has never been studied.

In this paper, we first survey 12 real-world OT malware samples and confirm that no existing OT malware employs anti-analysis checks, despite the clear opportunity to exploit OT-specific signals as sandbox fingerprints. Building on this finding, we introduce ShadowICS, a passive measurement framework that characterizes an OT-specific anti-analysis capability, trained on a novel taxonomy of OT features spanning temporal, physical-process, and network dimensions. Using lightweight machine learning models, ShadowICS quantifies how distinguishable real OT deployments are from simulations across multiple in-network vantage points, reaching up to 97.9% accuracy and revealing how often current sandboxes betray themselves under purely passive observation. Finally, we translate our findings into concrete mitigation strategies for sandbox designers, identifying temporal realism as the primary target for improvement.

Keywords: Industrial Control System · Sandbox · Malware.

1 Introduction

Industrial Control Systems (ICSs) operate essential infrastructure and industrial processes such as water treatment, water distribution, and smart grids. Unlike Information Technology (IT) systems, which handle data processing, communication, and business logic using general-purpose hardware and software, Operational Technology (OT) systems, such as those included in ICSs, directly

monitor and control physical processes through specialized hardware such as Programmable Logic Controllers (PLCs), sensors, and actuators.

ICSs are targeted by OT-specific malware capable of causing physical damage. For example, Stuxnet compromised a nuclear enrichment facility by manipulating PLC control logic [27], while Industroyer, disrupted power grids by replaying industrial protocol commands [15]. Unlike traditional IT systems, ICSs tightly couple software components with physical processes, leading to distinct attack surfaces and operational constraints.

ICS malware can be analyzed using static and dynamic analysis techniques. These approaches are valuable because they allow analysts to observe runtime behavior without reverse-engineering the whole malware logic [24]. For example, defenders rely on malware sandboxes, which are a virtualized environment, to safely study and dynamically interact with malware.

Advanced IT malware, like UpClicker and BaneChant [1], embeds anti-analysis techniques to detect virtualization artifacts, instrumentation traces, or inconsistencies in simulated systems [12, 52, 55]. Common IT anti-analysis techniques include inspecting environment-specific artifacts such as usernames, registry keys, or running processes; exploiting timing discrepancies introduced by virtualization; and identifying CPU or system-call inconsistencies that betray the presence of an instrumented environment.

On the other hand, anti-analysis techniques for OT malware remain underexplored, as existing research focuses on OT malware functionality, propagation, and impact [2, 11, 15, 27, 56]. To address this gap, we survey 12 state-of-the-art OT malware samples spanning 2010 to 2024 and examine their target and anti-analysis environment checks. We classify these checks into a novel four-class taxonomy of target checks (i.e., physical process, control logic, industrial network, industrial hardware) and find that none of the surveyed samples implements any anti-analysis check. Building on this finding, we also propose a new taxonomy of OT-specific anti-analysis techniques, organized along temporal, network, and physical-process dimensions.

We create ShadowICS, a passive measurement framework that quantifies how distinguishable simulated ICS deployments are from real ones by observing OT network traffic, thereby exposing which artifacts betray a simulated environment. We evaluate ShadowICS on 23 datasets covering both real and simulated ICS deployments, achieving up to 97.9% F1 score across all access levels using only 60 seconds of traffic, and maintaining above 81% F1 under the harder generalization setting. Our findings show how readily current simulated environments can be told apart from real ones under purely passive observation, and provide concrete guidance to sandbox developers in improving their tools.

Our contributions can be summarized as follows:

- We survey twelve advanced OT malware samples (Table 1) and establish, for the first time, that anti-analysis techniques for OT malware represent an unexplored surface, despite the opportunity to exploit OT-specific signals for sandbox detection.

- We design and implement ShadowICS, a passive OT measurement framework based on lightweight Machine Learning (ML) models trained on a novel taxonomy of OT-specific features spanning temporal, network, and physical process dimensions, characterizing the traffic artifacts through which simulated deployments become distinguishable from real ones.
- We evaluate ShadowICS across multiple access levels and datasets, achieving up to 97.9% accuracy with only 60 seconds of traffic, and we identify the structural weaknesses of current simulations and translate them into concrete hardening guidance for sandbox designers.

Availability. Due to concerns about dual-use aspects, we will not publicly release our code. The datasets’ metadata are reported in the appendix.

2 Background

OT systems ICSs are computing environments used to monitor and control industrial processes in sectors such as manufacturing, energy, and transportation [35]. Unlike traditional IT systems, ICS devices interact directly with the physical world through sensors and actuators, enabling automated supervision and control of processes such as temperature regulation, fluid flow, or machinery operation. These systems typically include PLCs, Human Machine Interfaces (HMIs), supervisory servers, and engineering workstations interconnected within an OT network [17].

Communication in ICS environments relies on dedicated industrial protocols designed for deterministic and real-time operation [36]. Among the most widely used are Modbus, S7Comm, and the Common Industrial Protocol (CIP). Modbus is a lightweight request–response protocol commonly used for register-based communication between PLCs and supervisory components, while S7Comm is a proprietary Siemens protocol for exchanging control logic and process data with SIMATIC PLCs. CIP, which supports protocols such as EtherNet/IP, is object-oriented and widely adopted in modern industrial automation systems [17]. These protocols enable the exchange of process measurements, control commands, and device-management information across the OT network.

ICS networks are often structured according to hierarchical architectures that separate enterprise IT systems from OT systems, including control and field devices, improving safety and operational determinism, as described in the Purdue model [66]. Such segmentation influences communication patterns and is commonly reflected in monitoring and analysis deployments.

OT Malware In the last few decades, the number of malware targeting OT systems has increased, including Stuxnet, Havex, and Industroyer [15, 26, 27]. An *ICS malware* includes a payload (i.e., the actually malicious part of the malware) that is specialized for manipulating physical processes, disrupting industrial operations, or conducting stealthy reconnaissance to steal industrial secrets. Despite a different payload, the OT malware typically includes similar components as an IT malicious software, as summarized in Figure 1. When the software

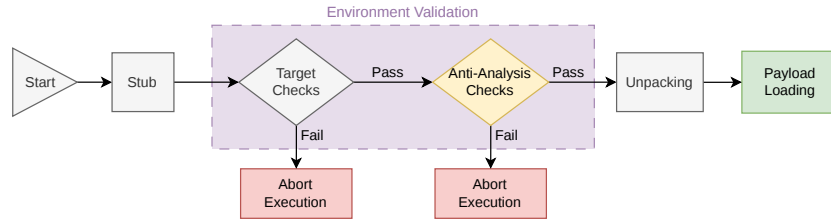


Fig. 1: Typical malware steps to load a payload: stub, environment checks, unpacking, and payload loading.

starts, a stub is loaded to perform preliminary operations and verify the conditions for executing the payload. If checks succeed, the malicious payload is unpacked and loaded into memory for execution.

Detecting and analyzing new malware is a difficult task that can be approached in different ways [20]. Static analysis examines malware without executing it. Analysts inspect binaries, strings, headers, and control flow to understand potential behavior. This approach is safe and fast, but it is limited when malware uses obfuscation techniques such as packing, encryption, control-flow flattening, or string encoding, which hide the real functionality and prevent accurate interpretation. To overcome such limitations, dynamic analysis observes malware while it is running [24]. By executing the sample, hidden behavior that is revealed only at runtime, such as unpacking routines, decrypted payloads, or network communication, can be captured. This makes dynamic analysis particularly useful for overcoming obfuscation, since many protections are removed or bypassed during execution.

To perform this safely, dynamic analysis typically runs malware inside controlled environments such as sandboxes, Virtual Machines (VMs), or system simulations. These environments isolate the malicious code from real systems while allowing analysts to monitor system calls, file changes, registry modifications, and network activity [12, 24]. In the OT field, simulations often also include emulating specialized hardware, protocols, and components to enable malware to run malicious code successfully [8, 56, 57]. However, discrepancies between simulated environments and real deployments may expose sandbox artifacts that advanced malware can exploit to evade dynamic analysis using anti-analysis techniques [1, 52, 58].

3 Motivation

Dynamic analysis has become essential in malware analysis because it reveals behavior that is difficult to recover through static inspection alone, particularly when malware employs packing, encryption, or runtime code generation [24]. In the OT domain, however, performing dynamic analysis is substantially more challenging than in traditional IT environments. OT malware often depends on

specialized industrial hardware, proprietary communication protocols, and interactions with physical processes that are difficult to reproduce faithfully in a sandbox or virtualized testbed [8, 17, 57]. As a result, current analysis environments rely on simulations that inevitably differ from real industrial deployments, creating artifacts that sophisticated malware could potentially exploit.

It is widely known that IT malware employs anti-analysis techniques to detect when it is executed in a sandbox, VM, or simulated environment and evade dynamic analysis [58]. Typical anti-analysis techniques include searching for VM artifacts, probing system configuration inconsistencies, or monitoring user interaction. A substantial body of work has investigated these behaviors in IT environments, providing surveys and empirical analyses of anti-analysis strategies and their implications for malware detection [1, 12, 55].

However, little attention has been devoted to *anti-analysis capabilities of OT malware* as existing research focuses on OT malware functionality, propagation, and impact [2, 11, 15, 27, 56]. As a result, it remains unclear whether current OT malware can recognize simulated industrial environments or which OT specific artifacts could enable such behavior. Understanding these capabilities is important because malware that detects an analysis environment could suppress or alter its malicious behavior, reducing analysts’ visibility into the actual payload. For example, a Stuxnet 2.0 malware variant might incorporate anti-analysis capabilities that identify simulated industrial environments by observing OT-specific network, timing, or physical-process artifacts, and withhold its malicious actions until deployed on a genuine target.

Motivated by this gap, we formulate and investigate three research questions:

- RQ1:** Are real-world and state-of-the-art OT malware samples using anti-analysis techniques?
- RQ2:** Can we create a taxonomy of anti-analysis techniques for OT malware and a related threat model?
- RQ3:** What are the challenges in designing, implementing, and evaluating an anti-analysis detection module tailored for OT malware?

4 OT Malware Survey

To answer **RQ1**, we studied the twelve critical OT malware samples listed in Table 1. We collected them from public resources, such as blogposts [2, 26, 64, 65], white papers [11, 15, 27, 40, 48, 61] and academic publications [33, 41, 56]. We focus on malware with a direct impact on industrial systems as they actively employ industrial protocols or specifically target industrial components. Our survey is not exhaustive but reflects the constraints of the OT malware ecosystem: OT-specific samples are rare, and their binaries or source code are seldom publicly available. We selected publicly documented malware and relevant academic proofs-of-concept from 2010 to 2024 that directly impact industrial processes, and accordingly report no evidence of anti-analysis checks in the surveyed public samples and reports, rather than their universal absence. We gathered the identified ICS malware in a GitHub repository to allow others to contribute to

Table 1: Survey of twelve real-world OT malware. **P**: malware provenience from the wild  or academic . In the last three columns, we use  if the malware is publicly available (**S**), uses target checks (**T**), or anti-analysis checks (**A**); otherwise, we use .

ID	Name	Year	Target	ICS Protocol(s)	P	S	T	A
STX	Stuxnet [27]	2010	Nuclear facilities	S7Comm				
HAV	Havex [26]	2013	Various ICS	OPC-UA				
BE3	BlackEnergy3 [61]	2015	Smart grid	-				
IRN	Irongate [48]	2015	Various ICS	S7Comm				
IN1	Industroyer [15]	2016	Smart grid	IEC61850/101/104,OPC-DA				
HAR	Harvey [33]	2017	Power plant	-				
TRI	Trisis [40]	2017	Safety Instruments	TriStation				
CLK	CLIK [41]	2019	Various ICS	-				
IN2	Industroyer2 [65]	2022	Smart grid	IEC104				
PIP	Pipedream [11]	2022	Various ICS	Modbus,OPC-UA,CodeSys				
FRG	FrostyGoop [2]	2024	Heating control	Modbus				
FUX	Fuxnet [64]	2024	Various ICS	Meter-Bus				

the collection⁴. Next, we introduce the terminology we use to classify environment checks, analyze the checks employed by each malware, and organize target checks into a novel four-class taxonomy.

4.1 Terminology

We present the survey terminology using Figure 1 as a reference. A *target* check verifies that the malware is executing in an environment that satisfies the requirements for successful payload execution. For example, checking for the presence of specific PLC constitutes a target check when their availability is necessary for correct malware operation. In contrast, an *anti-analysis* check aims to determine whether the malware is running in a simulated environment, such as an OT sandbox, honeypot, or digital twin. In this case, the mere presence of a PLC is insufficient; the malware must also assess its realism to ensure it is not a decoy. Both checks, which can be passive or active, are performed before the payload is delivered. An *environment* check is either a target or an anti-analysis check.

4.2 Environment Checks

We reviewed the environment checks used by twelve state-of-the-art OT malware and compiled Table 1. The table includes samples from 2010 to 2024, targeting critical infrastructure such as nuclear plants, smart grids, power plants, and safety industrial systems. The malware payloads use various industrial protocols, including S7Comm, OPC-UA, Modbus, and Meter-Bus. Two malware samples

⁴ <https://github.com/donadelden/awesome-ics-malware>

Table 2: ICS target checks on known OT malware.

Category ↓	Malware ID →	STX	HAV	BE3	IRN	IN1	HAR	TRI	CLK	IN2	PIP	FRG	FUX
Physical Process		✓	✓		✓		✓						✓
Control Logic					✓	✓	✓	✓	✓				
Industrial Network		✓	✓			✓		✓		✓	✓	✓	
Industrial Hardware		✓			✓	✓	✓	✓	✓		✓	✓	

are academic proof-of-concepts [33, 41], which we include because they represent innovative frameworks that attackers may implement—or may already be implementing—in novel OT malware.

As shown in Table 1, we find *no anti-analysis checks* (A column) and several target checks. To the best of our knowledge, target checks have not been systematically surveyed and classified. Providing such a classification is important for establishing a common terminology, enabling consistent comparison across malware samples, and supporting the identification of trends and gaps that can inform both defensive strategies and future research. We classify the target checks into four classes: i) *Physical Process*, ii) *Control Logic*, iii) *Industrial Network*, and iv) *Industrial Hardware*. Next, we describe each class in detail, and we map their usage to the malware in Table 2.

Physical Process The presence of a physical process is a central feature of an ICS and is used by certain OT malware to check for a specific target. In particular, malware often queries devices to get sensor measurements and verify the presence of a physical process. The Havex malware, for instance, attempts to query OPC server objects connected to a physical process to verify whether the target is an industrial system [54]. Stuxnet goes further by treating the physical process state as an activation condition. It reads the rotational frequency reported via Profibus and only proceeds if the value falls between 807 Hz and 1210 Hz , a range characteristic of gas centrifuge operation [27]. Instead, Irongate adopts an implicit approach: its payload records 5 seconds of live PLC process IO traffic and then replays it to the HMI as a deception layer. This step presupposes an active physical process producing sensor values [48].

Control Logic Control logic properties concern the structure and availability of the programs executed by industrial controllers (e.g., ladder logic developed by engineers). Some malware relies on downloading control logic from PLCs, modifying it, and uploading a compromised version [9]. However, the mere presence of control logic does not guarantee consistency with a target industrial process [19]. Irongate searches for a specific DLL and intercepts specific calls used by the engineer-written program to exchange data with the simulated controller [37, 48]. The payload, therefore, cannot operate unless the target control logic interface is present and active.

Trisis queries the Safety Instrumented System controller for its program state and key state prior to any manipulation, and reads controller memory to locate the existing safety logic before uploading a replacement ladder diagram [7, 39]. This read step ensures the malware is operating on a live PLC. CLIK goes furthest in this regard: it first downloads the binary control logic from the target PLC, decompiles it into Instruction List (IL) source code, injects malicious rungs into the existing program structure, and only then re-uploads the modified binary, making the pre-existing engineer-written logic a necessary input to the attack itself [41].

Industrial Network Industrial network checks are implemented by OT malware to ensure that the compromised host resides within an ICS. A frequent technique is network scanning to locate industrial devices, ranging from broad subnet enumeration on industrial ports (44818, 102, 502) in Havex to more targeted discovery mechanisms, such as the UDP broadcast probe over port 1502 included in the Trisis codebase for locating Triconex controllers [7], or multi-protocol scans in Pipedream [11].

Malware also performs protocol compliance verification, for instance by validating specific protocol handshakes or signatures, as seen in Industroyer2’s IEC 104 state machine sequence, which requires a valid `STARTDT` exchange before issuing commands [56], confirming that the target implements the expected industrial communication stack. While many checks in this category are done, their goal is always to identify the correct target, not to detect analysis environments.

Industrial Hardware Industrial hardware checks are employed by OT malware to verify that the target device matches the specific hardware configuration for which an exploit or payload was designed. Stuxnet is the best-documented example, performing multi-step hardware fingerprinting before activating its payload. It queries the PLC CPU model identifier and reads the System Data Blocks to verify that the expected Profibus communications module and specific frequency converter drives are present [27, 45].

Trisis gates its privilege-escalation exploit on firmware compatibility: the exploit targets the Triconex MP3008 module on a specific firmware version range, and the injector performs a preliminary memory probe to confirm the vulnerable code path is available before committing to the full payload injection, aborting on non-matching devices [7, 40]. Pipedream takes a coarser but structurally analogous approach: its dedicated modules scan and identify Schneider Electric and Omron PLCs by vendor and model, using protocol-specific interactions to confirm device identity [11, 23]. Across these cases, hardware identification serves both as a targeting filter and as an exploit-compatibility check, ensuring the payload is deployed on devices whose architecture and firmware make it viable, but without the goal of detecting analysis environments.

Summary Our survey of twelve OT malware samples reveals a consistent pattern: *all surveyed samples rely exclusively on target checks, and none of the sur-*

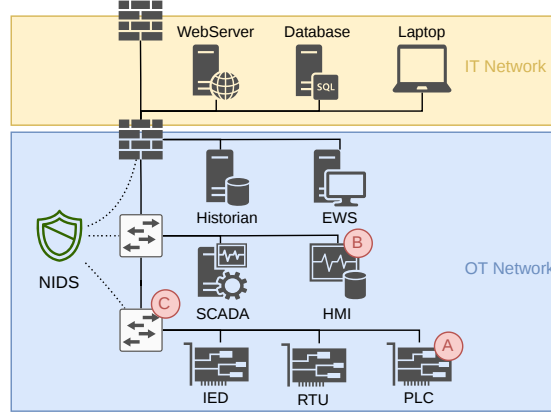


Fig. 2: Threat model. The attacker can install OT malware on a PLC (A), HMI (B), or industrial switch (C). They can observe network traffic between two or more devices (Low, Medium, High network access level).

veyed samples implements anti-analysis checks (Table 1, column A). The four target-check categories we identified confirm that OT malware is already process-aware and network-aware, yet this awareness has never been directed toward detecting analysis environments. This absence is significant: it shows that OT-specific anti-analysis techniques represent a largely unexplored attack surface, leaving a concrete gap in both offensive understanding and defensive preparedness. This finding directly motivates **RQ2** and **RQ3**: what does a taxonomy of OT-specific anti-analysis checks look like, and how effectively can such checks be implemented in practice? In Section 5, we address these questions by presenting ShadowICS, a novel passive measurement framework that demonstrates, for the first time, that industrial network traffic alone is sufficient to distinguish simulated ICS environments from real ones.

5 ShadowICS

Next, we present ShadowICS, a new passive measurement framework that leverages network traffic analysis to quantify how distinguishable a simulated OT deployment is from a real one. By characterizing the traffic artifacts through which a sandbox or digital twin betrays itself, ShadowICS captures the same properties that an OT malware sample, such as those listed in Table 1, could exploit to detect an analysis environment, so that defenders can prioritize remediation. Next, we present ShadowICS’s threat model, architecture, classification features, and implementation details.

5.1 Threat Model

Our system model includes a generic ICS comprising an IT and OT part. As shown in Figure 2, the ICS is compromised by an OT attacker targeting a PLC, an HMI, or an industrial switch. Initial compromise may occur through multiple vectors, including compromised engineering workstations, remote maintenance connections, infected removable media, or by pivoting from the enterprise IT network into the OT environment, as demonstrated by previous attacks on industrial systems [15, 27]. The compromised host determines the access level: a PLC-hosted instance sees mostly its own flows, whereas medium and high visibility require compromising an HMI, historian, gateway, switch, or a mirrored link. We focus on a passive observer seeking to detect whether it is inside a sandbox by observing industrial network traffic. We consider this vantage point because it is passive, stealthy, and generic, and therefore represents the hardest case for a defender to detect and counter. We consider attackers with three network access levels:

- (1) **Low**: attacker sees traffic between two components, like a PLC and an HMI.
- (2) **Medium**: attacker sees traffic between multiple components, like the HMI and all devices reporting to it.
- (3) **High**: attacker sees all traffic in a (sub)network, like all the link layer traffic passing through an industrial switch.

The attacker uses such an approach to ensure that it is running on a genuine system rather than a simulation used for dynamic analysis [8, 43, 57]. In this context, active probing is risky: generating anomalous network traffic (e.g., sending a register read) may trigger Network-based Intrusion Detection System (NIDS) [36], device-level artifact checks may be caught by Host-based Intrusion Detection Systems (HIDSs) [6, 49], and many architectural checks require root access that is difficult to obtain prior to full payload execution. Communicating with the outside world may also be challenging due to the airgapped network of many OT networks in ICSs. Therefore, the attacker will *passively* collect traffic received by the compromised device and process it locally to identify the presence of simulated devices. Because this check runs before any payload is executed, the traffic the observer sees is benign traffic by construction. We do not consider an *active* attacker capable of probing the network or leveraging device artifacts; such capabilities would only increase the information available and thus make simulated environments even easier to distinguish, so the passive setting we study is a conservative lower bound.

5.2 Architecture Features

Figure 3 presents the ShadowICS architecture. Our system takes network traffic as input, extracts relevant features, selects the best features, and, with a certain probability of error, classifies whether each device it sees in the traffic is simulated. Each classification probability can also be combined to determine whether the entire system is simulated.

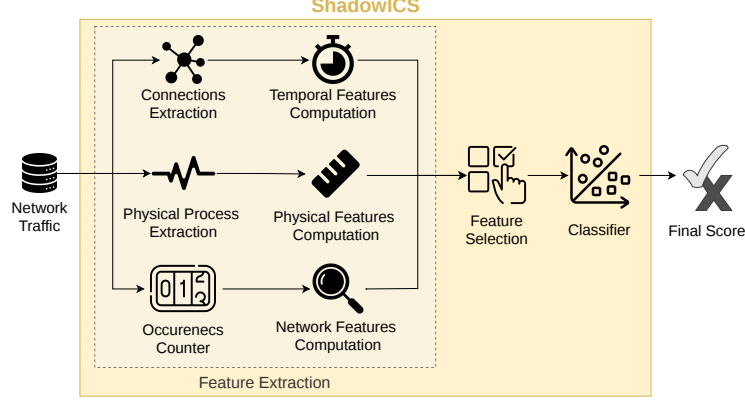


Fig. 3: ShadowICS architecture.

To answer **RQ2**, we created, after cleanup, 86 OT features (89 for high access), which are complementary to the target checks we survey in Section 4 and to the IT ones. A full list is reported in the Appendix (Table 7). They are divided into three classes: i) Temporal, ii) *Network*, and iii) *Physical* features.

Temporal Features Industrial devices such as PLCs operate according to a fixed *scan cycle*: the recurring sequence of reading sensor inputs, executing the control program, and writing actuator outputs. This cycle repeats at a deterministic rate, typically in the range of a few to tens of milliseconds, and governs the timing of all outgoing network traffic. As a result, communication patterns in a real ICS are highly periodic and coupled to this hardware-level schedule. Simulators, by contrast, often introduce artificial regularity or excessive jitter, both of which deviate from real hardware. Building on prior work that leverages scan-cycle periodicity for device fingerprinting [3], we extracted 23 temporal features organized around four timing primitives illustrated in Figure 4, each computed as mean and standard deviation using both sniff timestamps and TCP timestamps where available.

The most direct expression of the scan cycle is the time separating consecutive messages of the same type on a given connection, computed between pairs sharing the same source-destination pair [60]. Specifically, *Req2Req* captures how often the supervisory component polls the field device, *Res2Res* captures the response periodicity on the device side, and *Res2Req* represents the idle interval between two polling rounds. All three are tightly constrained in real deployments and difficult to reproduce accurately in simulation. When the TCP timestamps option is enabled, we derive a parallel set of the same six features from the TCP-layer timestamps, which are less affected by the sniffing point. We also extract the *Req2Res* delta, i.e., the time between a request and the corresponding response, which reflects the device’s processing latency.

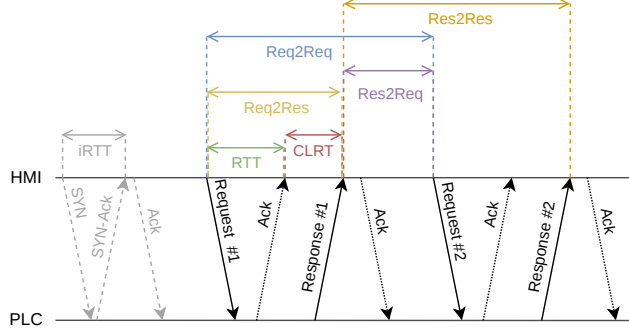


Fig. 4: Measurements connected to PLC scan cycle employed by ShadowICS to extract temporal features.

We additionally compute the Cross-Layer Response Time (CLRT), i.e., the interval between an application-layer response and its transport-layer acknowledgment, which is proportional to the remaining time in the current scan cycle [30]. Real hardware produces a narrow, consistent CLRT distribution, whereas simulators tend to collapse this value to zero or introduce high variance. The Round Trip Time (RTT), i.e., the time between a request and its transport-layer acknowledgment, is similarly tightly bounded on real hardware but destabilized by the scheduling noise of general-purpose simulators.

Finally, the initial RTT (iRTT), measured during the TCP three-way handshake before any application-layer exchange, provides a clean baseline that reflects only the host’s network stack latency; prior work shows that TCP handshake parameters, such as window scaling, carry strong fingerprinting potential for industrial devices [18]. Finally, we also include a protocol label as a feature, since different industrial protocols rely on distinct communication stacks and may impose different timing constraints.

Simulated environments tend toward one of two failure modes: either they are *too regular* (perfectly constant deltas with zero variance, which no real hardware does) or *too noisy* (high variance introduced by a general-purpose OS scheduler), both of which are captured by the mean and standard-deviation feature pairs.

Physical Features A characteristic unique to ICSs with respect to IT systems is the presence of one or more physical processes that the control logic supervises and acts upon. Measurements collected from such processes are shared between devices: for instance, PLCs transmit sensor readings and actuator statuses to the HMI to update its graphical depiction of the system state. ShadowICS passively observes this traffic to reconstruct the evolution of each physical process and extracts 60 physical features organized into three groups.

The first group captures *base process statistics*, such as normalized standard deviation and a binary flag indicating whether the process value is constant over the observation window. Ideally, a steady process is a strong indicator of a

mock device, as real physical processes exhibit variability due to environmental conditions and control actions.

The second group characterizes the *statistical and structural properties* of the process signal, motivated by the fact that real physical processes exhibit noise and dynamics that are non-trivial to emulate [4, 22]. We extract variance, SNR, skewness, kurtosis, and permutation entropy to capture the energy, distributional shape, and irregularity of the signal; the number of peaks, longest runs above and below the mean, and the number of mean crossings capture its temporal structure; and the Augmented Dickey-Fuller p-value provides a stationary test. These features are also extracted from an *estimate of the sensor noise*, obtained by subtracting a filtered signal from the original in a low-variability window of x samples lying within $[\mu - \epsilon, \mu + \epsilon]$ [22]. From this noise signal, we extract its mean, standard deviation, median, root mean square, binned entropy, mean crossings, counts above and below the mean, and spectral descriptors via FFT.

The third group captures *cross-process relationships* via the maximum and minimum cross-correlation and cross-coherence between process pairs. In a real ICS, actuating one part of the system causally affects others, so low cross-process coupling is a strong indicator of independent, simulated processes.

Network Features OT networks differ from classical IT networks in their fixed topology, deterministic communication patterns, and the specific characteristics of the connected devices. We extract 40 network features in the low- and medium-access scenarios and 52 in the high-access scenario, organized into three groups.

The first group captures *device fingerprinting* signals. Every MAC address begins with an Organizationally Unique Identifier (OUI), registered by manufacturers to identify their products and stored in the IEEE OUI database⁵. We compute the rate of MAC addresses belonging to known ICS vendors and the share of addresses that clearly look fake (e.g., starting with AA:BB:CC or 11:22:33). While MAC addresses are easily spoofable, the absence of any PLC vendor in the traffic is a strong indicator of simulation. We also compute the number of unique TTL values across devices: in real networks, heterogeneous hardware yields varied TTLs, whereas simulated environments tend to exhibit uniform TTLs, suggesting cloned or templated devices [18].

The second group targets *traffic artifacts*. OT networks are deterministic and rarely exhibit broadcast or management traffic such as ARP, ICMP, DNS, or DHCP. We therefore count the rates of such protocols, along with UDP traffic, discovery packets, and duplicate IP warnings, any of which may betray the presence of simulation tooling. We also track the IP Don't Fragment flag, which legacy PLC firmware may not set, and monitor packet rates at both global and per-device levels.

The third group extracts *TCP connection features*. Long-standing TCP connections are typical in real ICSs [14], whereas simulators often establish a new connection per request, resulting in a higher rate of SYN and FIN flags and fewer reused sessions. We compute the rates of TCP control flags, the session reuse

⁵ <https://standards-oui.ieee.org/>

rate, and the count of TCP anomalies, keep-alives, and advanced options such as SACK and timestamps. For payload characterization, we extract the mean, standard deviation, maximum, and mode of both the overall and industrial-layer TCP payload sizes, as well as the bytes sent since the last PSH flag. UDP features are limited to the rates of industrial-layer UDP traffic and other UDP packets, given the small number of meaningful fields in the UDP header.

5.3 Classification

Features are extracted from each connection between two devices. In the low- and medium-access scenarios, one of the two endpoints is always the host on which ShadowICS is deployed; in the high-access scenario, both endpoints can be identified from intercepted traffic. Since not all features contribute equally to discrimination, a feature selection phase retains only the most informative features, reducing the risk of overfitting.

For each connection, the classifier outputs the probability that the device sharing physical process information is simulated. In the low-access scenario, this per-connection score is used directly. In the medium and high-access scenarios, where multiple connections are observable, a voting mechanism combines the per-connection probabilities into a single score that reflects the realism of the overall ICS.

5.4 Implementation

The implementation of ShadowICS is consistent with the architecture provided in Figure 3, and consists of two sequential steps. The first step parses each packet in real time (or from a PCAP capture) using a series of if-else statements, without storing the analyzed packets. It extracts the information needed for feature computation: protocol counters, register values for physical process reconstruction, and request-response tracking for temporal measurements. To mimic a realistic deployment, features are extracted from a fixed 60-second traffic window, regardless of dataset size.

The second step computes the final features from the data collected in the first step: for example, autocorrelation of physical processes and statistics on TCP traffic. Features whose value depends on the number of processed packets are normalized to rates, enabling comparison across datasets and connections with different traffic volumes. Although some protocol-specific adaptations are required, such as physical process extraction and request/response definitions, the outputs are always protocol-agnostic.

After an initial variance-thresholding step that removes near-constant features, the pipeline yields 86 features in the low- and medium-access scenarios and 89 in the high-access scenario. Final feature selection is then performed using a Random Forest (RF) importance score or ANOVA, depending on the target classification model.

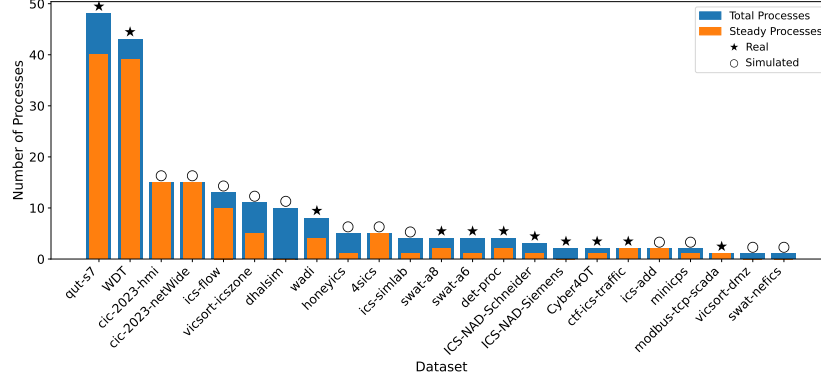


Fig. 5: Number of physical processes included in each dataset.

6 Evaluation

We evaluate ShadowICS on the datasets presented in Section 6.1, and Section 6.2 discusses classification performance across access levels and feature subsets, as well as the final results, thereby providing a response to **RQ3**.

6.1 Dataset

Collecting suitable data for testing ShadowICS is challenging due to the scarcity of compatible ground-truth datasets in the ICS security literature and the high cost of building or accessing real industrial testbeds [17]. To address this, we collected datasets from two sources: existing public captures from the literature covering both real and simulated environments, and additional captures we generated from virtualized testbeds shared by the community [8, 25, 53, 57]. An overview of all datasets is provided in Table 6 in the Appendix.

Each dataset is assigned one of two ground-truth labels. A dataset is labeled *real* if all the industrial devices generating the captured traffic are physical hardware deployed in an operational or research testbed, and *simulated* if all devices are software-emulated or virtualized (e.g., MiniCPS [8], NEFICS [56]). Datasets in which physical hardware and software emulation coexist were excluded, as the mixed composition makes it impossible to assign a verifiable ground-truth label to each individual device. Labeling is based on the testbed description provided by the dataset authors; where documentation was ambiguous, we conservatively excluded the dataset. If a fine-grained labeling is available, ShadowICS should theoretically work even with hybrid datasets containing both real and simulated devices; we left this analysis as a future work due to the lack of data.

Starting from the broad collection surveyed in [17], we identified 23 datasets that satisfy all of the following requirements. First, captures must include benign traffic only, or contain clearly labeled attack-free intervals, since malicious

Table 3: Scores in different access levels with the best 20 features using a RF.

Access	Support	Precision	Recall	F1	Accuracy
Low	75	0.9733	0.9733	0.9733	0.9733
Medium	79	0.9771	0.9771	0.9771	0.9771
High	23	0.9800	0.9792	0.9791	0.9792

interactions alter communication patterns in ways that would confound our feature extraction. Second, datasets must be provided as raw PCAP files: captures pre-processed into CSV or feature matrices were excluded as they prevent full feature extraction. Third, the testbed description must be sufficiently detailed to assign an unambiguous ground-truth label, as discussed above. Fourth, the captured traffic must involve at least one active physical process, since process-derived features (Section 5.2) require observable sensor or actuator values in the payload.

Finally, the traffic must use at least one of the three industrial protocols covered by the current prototype (S7Comm, Modbus, or CIP), though the feature pipeline is largely protocol-agnostic and extending support to additional protocols requires minimal effort. Figure 5 shows the number of physical processes recoverable from each dataset’s traffic, with a superimposed bar indicating the share of *steady* processes (i.e., those exhibiting a constant value throughout the observation window).

6.2 Results

Final Performances Table 3 reports the average results of 5-fold stratified cross-validation (80/20 split) for each access level, using the top-20 features selected by RF importance. ShadowICS achieves consistently high classification performance across all three access levels, with F1 scores ranging from 97.3% (low) to 97.9% (high). The near-perfect alignment between precision and recall reflects a well-balanced discrimination between real and simulated devices, with few false positives or false negatives in either class. Overall, these results confirm that passive network monitoring alone is sufficient for reliable environment discrimination across realistic attacker positions. We treat standard cross-validation as an upper bound, because samples from the same dataset may share testbed artifacts; Leave One Out Cross-Validation (LOOCV), discussed below, is the stricter estimate for unseen-environment generalization.

Ablation Study First, we conducted a simple *baseline study* to assess the performance of a threshold applied to the best feature in each category. In a low-access scenario, results are unsatisfactory, yielding accuracies of 0.55, 0.52, and 0.13 when selecting only the feature with the highest variance in the network, physical, and temporal categories, respectively. Then, to understand which feature families drive classification, we performed an ablation study evaluating each feature type in isolation using a simple Decision Tree (DT) model. Results are shown

Table 4: Scores with single feature category. # indicates the number of features.

<i>Access</i> →	Low			Medium			High		
Type	#	F1	Accuracy	#	F1	Accuracy	#	F1	Accuracy
Temporal	15	0.9595	0.9600	15	0.9536	0.9542	15	0.9371	0.9375
Physical	49	0.7828	0.7867	49	0.7823	0.7863	49	0.7474	0.7500
Network	21	0.9598	0.9600	21	0.9540	0.9542	24	0.9164	0.9167

in Table 4. Temporal and network features, when considered individually, yield F1 scores above 0.91 at every access level, confirming that either family alone is sufficient for strong discrimination. A possible explanation for the lower performance when using only physical features is the limited sampling granularity of the physical-process reconstruction, which relies solely on samples transmitted over the network. This limits the quality and reliability of the noise-based statistics extracted from the processes. In a scenario where the attacker has direct access to higher fidelity process measurements, the contribution of this feature class would even increase further [4, 22].

Feature Importance Figure 6 reports mean SHAP [47] values for the low-access scenario, averaged across five splits, to identify the attributes with the greatest discriminative power. Identifying the most influential features is useful for a sandbox designer who can now target the most fingerprintable aspects of their emulation.

Temporal features dominate the ranking. Deviations in the periodicity of request–response cycles, specifically the standard deviation of *Res2Req* and *Res2Res* inter-arrival times derived from TCP timestamps, are the top two contributors. This is consistent with the strict scan-cycle scheduling of real PLC hardware: simulators either reproduce these intervals too perfectly (zero variance) or too erratically (excessive jitter from a general-purpose OS scheduler), making both failure modes easy to detect. Implementation-specific differences in TCP timestamp handling further amplify the signal.

Network features form the second most impactful group, with average TCP payload size, acknowledgment count, control flags count and mean TTL all contributing meaningfully. These parameters reflect low-level implementation details of the underlying OS, firmware, and protocol stack, and are difficult to reproduce consistently across heterogeneous simulated devices. Physical-process features, despite their lower aggregate importance (consistent with Table 4), contribute via noise-derived statistics, such as the median and mean of extracted sensor noise, aligning with prior work on noise fingerprinting in cyber-physical systems [4, 22].

Model Selection Figure 7 shows how F1 varies with the number of selected features for six classifiers in the low-access scenario. Tree-based ensemble models (Extreme Gradient Boosting (XGB), RF, and DT) stabilize above $F1 = 0.9$ with

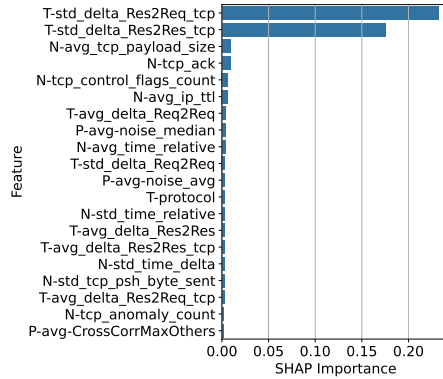


Fig. 6: Mean SHAP values of the most important features in a low access scenario.

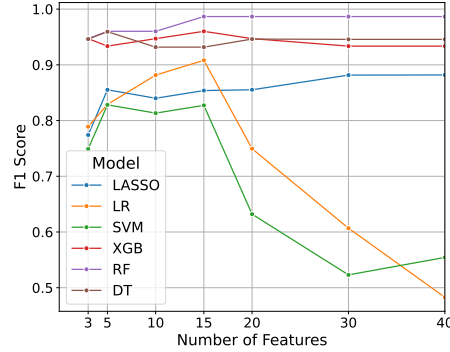


Fig. 7: F1 score variance of model vs number of features in a low access context.

as few as 10 features and remain robust as features increase. Linear models (Logistic Regression (LR), Support Vector Machine (SVM)) reach comparable performance but do so more efficiently with a smaller feature budget, plateauing around 10–15 features. All models achieve strong performance at 20 features, confirming that the feature-selection threshold adopted throughout this evaluation is conservative rather than critical.

LOOCV To approximate a realistic deployment, we tested ShadowICS under an LOOCV strategy, in which the model is trained on all but one dataset and tested on the held-out dataset. This emulates an attacker who trains on publicly available captures before targeting an unknown ICS. We also adopt this strategy because ICS datasets may share testbed-specific artifacts, such as capture-point configuration or OS and network stack defaults, which could introduce correlation across traces and inflate standard cross-validation scores if similar artifacts appear in both training and test folds. Results are shown in Table 5.

Despite the substantially harder generalization requirement, classification performance remains above 0.81 F1 across all access levels. The gap with respect to standard cross-validation (≈ 0.15 F1) is expected, given that LOOCV tests on entirely unseen environments; the fact that accuracy remains strong indicates that ShadowICS captures structural properties of real vs. simulated deployments that are not simulation-specific.

Computational Requirements Feature computation is lightweight and suitable for real-time execution. Temporal features require only delay measurements and minimal session buffers, while network features rely on counters with average and standard deviation computed before experiments. Physical process features can also be extracted online once sufficient packets are received. As shown in Figure 7, fewer than 20 features are sufficient for most models to achieve good

Table 5: Scores using LOOCV.

Access	Support	Precision	Recall	F1	Accuracy
Low	75	0.8291	0.8133	0.8160	0.8133
Medium	79	0.8268	0.8228	0.8227	0.8228
High	23	0.8729	0.8696	0.8696	0.8696

accuracy, resulting in limited computational overhead for testing ShadowICS. Time-wise, only 60 seconds of traffic were used in these experiments, which shows that even brief sandbox exposure is enough to be identified, which leaves sandbox operators very little margin. Analysis of other time windows is left as future work. Our findings also demonstrate that the trained model remains small. Models such as DT and Least Absolute Shrinkage and Selection Operator (LASSO) require less than 5kB of storage, while RF models typically require around 30kB.

7 Mitigation strategies

The results suggest that improving temporal realism should be a primary objective for sandbox and testbed designers. Since timing-related features play a central role in the classification, accurately reproducing scheduling behavior, scan-cycle periodicity, and processing delays is more effective than focusing only on high-fidelity physical-process simulation. Small discrepancies in timing patterns can introduce artifacts that are easily captured by passive observation, making temporal inconsistencies a significant source of fingerprinting.

Network-level characteristics also require careful consideration. Parameters such as TTL values, TCP control flags, acknowledgment patterns, and payload size distributions often reflect implementation details of the underlying operating system and protocol stack. Ensuring consistency across these elements can reduce the number of distinguishable artifacts. In particular, avoiding uniform configurations and reproducing realistic device-to-device variability can further limit the effectiveness of passive classification.

In contrast, perfect replication of the physical process appears less critical under the threat model considered. Deviations in process dynamics are generally more difficult to detect through network-only observation, and therefore contribute less to fingerprintability. Nevertheless, introducing realistic noise and variability in process values may still reduce detectability, as prior work suggests [4, 22]. Such measures can complement improvements in temporal and network realism without requiring fully accurate physical modeling.

8 Related Works

IT Evasive Malware The ability of malware to detect analysis environments and suppress malicious behavior has long been documented in traditional IT

settings [1, 52]. Prior work categorizes evasion techniques based on observable artifacts left by virtualization and sandbox instrumentation [12]. Common approaches include inspecting environment-specific indicators such as usernames, files, or running processes; exploiting timing discrepancies introduced by virtualization; and identifying CPU or system-call inconsistencies. Other techniques attempt to infer user activity through so-called reverse Turing tests, for instance by monitoring mouse movements, interacting with the user, or analyzing filesystem usage patterns. Network-level inconsistencies and simplified device emulation have also been leveraged to identify simulated environments.

Subsequent studies confirmed the widespread adoption of such techniques in real-world malware. For example, Sharma et al. [58] analyzed more than 4,000 advanced persistent threat samples and found anti-analysis capabilities in nearly all of them. These mechanisms typically target general-purpose computing environments and rely on artifacts specific to desktop, mobile, or browser-based platforms [42, 55]. In contrast, little attention has been paid to evasion strategies tailored to ICS deployments, where attackers can exploit characteristics of the physical process and industrial communication patterns.

ICS Malware Analysis Several works have analyzed known ICS malware to characterize their capabilities and operational objectives [10, 35, 51]. These studies highlight the increasing sophistication of industrial threats, including stealthy manipulation of control logic, firmware-level compromises, and coordinated attacks against physical processes. For instance, Harvey [33] introduces a PLC rootkit that modifies control commands while maintaining consistent sensor readings, whereas CLIK [41] demonstrates automated manipulation of ladder logic. Other works present proof-of-concept attacks, such as MadIoT [59, 62], self-learning malware targeting cyber-physical infrastructures [16], and ladder-logic-based payloads [29, 34]. Despite demonstrating advanced process-aware behavior, these contributions primarily focus on achieving stealthy manipulation of industrial systems rather than evading analysis environments. To the best of our knowledge, none of the documented ICS malware explicitly incorporates mechanisms to detect sandboxing or simulation.

ICS Virtualization and Analysis Environments Analyzing ICS malware requires testbeds that can reproduce industrial protocols and physical processes. To address this challenge, researchers have proposed virtualized, hybrid, and high-interaction environments that emulate control components and plant dynamics [17]. Examples include MiniCPS [8], a Mininet-based framework for cyber-physical simulations, and NEFICS [56], a reconfigurable environment for studying malware such as Industroyer. In parallel, ICS honeypots have evolved from low-interaction decoys to complex, high-interaction platforms that aim to realistically mimic industrial deployments [38, 46], including accurate simulation of physical process noise [22]. These systems are designed for both malware analysis and the observation of attacker behavior in controlled settings. Our paper’s novelty does not primarily reside in the individual fingerprints, but in

their composition into a passive OT anti-analysis measurement framework with a taxonomy and different access-level evaluations.

However, accurately reproducing device behavior, communication patterns, and physical process dynamics remains challenging. Prior work has shown that honeypots and simulated environments may expose inconsistencies that allow their identification, for example, through protocol fuzzing or behavioral analysis [63, 67]. While such studies focus on external attackers probing decoy systems, similar limitations also affect sandboxed execution environments used for malware analysis, potentially enabling detection from within the compromised system.

9 Conclusions

We investigated how distinguishable real and simulated ICS deployments are under purely passive network observation, and introduced ShadowICS, a measurement framework that quantifies this distinguishability solely from network traffic, reaching up to 97.9% accuracy. The results show that current simulated environments can be told apart from real ones across multiple access levels even from the most constrained vantage point that avoids active probing. This exposes a concrete weakness in today’s analysis environments: if dynamic analysis relies on simplified or imperfect simulations, evasive OT malware able to detect them could plausibly go unnoticed in the wild.

These findings highlight structural limitations in current ICS sandboxes and testbeds. In particular, temporal characteristics emerge as key distinguishing factors, indicating that improving timing realism should be prioritized when designing analysis environments. By identifying these weaknesses and proposing mitigation strategies, this work aims to support defenders in strengthening sandbox robustness and improving the detection of previously unseen OT-specific threats.

Ethical Considerations The primary goal of this work is to expose weaknesses in existing analysis environments and inform defensive improvements. The discussion of mitigation strategies provides concrete guidance for sandbox designers to reduce fingerprintability, while ShadowICS enables testing the performance of new sandboxes. Increasing awareness of these evasion vectors contributes to more resilient analysis pipelines and may facilitate the discovery of OT-targeted malware that would otherwise evade detection. To avoid potential dual-use issues, we decided not to release ShadowICS publicly.

Limitations and Future Work This work presents a few limitations. First, the proposed approach relies solely on features extracted from network traffic. This design aligns with a stealthy threat model and ensures broad applicability, but additional information obtainable through device-level inspection, such as architectural or firmware-specific characteristics, could enhance discrimination capabilities. Exploring such features would require access to a heterogeneous set of industrial devices and is left for future work.

Second, defining a strict ground truth for ICS datasets is inherently challenging. Modern testbeds often combine simulated components with real hardware, resulting in hybrid environments that do not fit neatly into binary categories. While this may introduce some degree of label ambiguity, it also reflects realistic deployment scenarios and therefore does not undermine the evaluation’s practical relevance. Finally, the evaluation covers a representative, though not exhaustive, set of industrial infrastructures and protocols. Although the considered features are largely protocol-agnostic and results remain consistent across datasets, extending the analysis to additional environments would further validate the approach’s generality.

Acknowledgments Denis Donadel and Massimo Merro have been partially supported by the SERICS project (PE00000014) under the *MUR National Recovery and Resilience Plan*, funded by the EU - NextGenerationEU. Daniele Antonioli has been partially supported by the French National Research Agency (ANR) under the France 2030 label, with the REV (ANR-22-PECY-0009) and NF-HiSec (ANR-22-PEFT-0009) research grants.

References

1. Afanian, A., Niksefat, S., Sadeghiyan, B., Baptiste, D.: Malware dynamic analysis evasion techniques: a survey. *ACM Computing Surveys (CSUR)* **52**(6), 1–28 (2019)
2. Ahlers, C., O’Meara, K.: Impact of FrostyGoop ICS Malware on Connected OT Systems. Tech. rep., Dragos Inc (2024)
3. Ahmed, C.M., Ochoa, M., Zhou, J., Mathur, A.: Scanning the cycle: Timing-based authentication on plcs. In: *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. pp. 886–900 (2021)
4. Ahmed, C.M., Ochoa, M., Zhou, J., Mathur, A.P., Qadeer, R., Murguia, C., Ruths, J.: Noiseprint: Attack detection using sensor and process noise fingerprint in cyber physical systems. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. pp. 483–497 (2018)
5. Ahmed, C.M., Palleti, V.R., Mathur, A.P.: Wadi: a water distribution testbed for research in the design of secure cyber physical systems. In: *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*. pp. 25–28 (2017)
6. Alla, O., Mocanu, S.: Host-based intrusion detection for industrial control systems. In: *2025 IEEE 11th International Conference on Network Softwarization (NetSoft)*. pp. 265–268. IEEE (2025)
7. Antiy CERT: Technical analysis of industrial control malware TRISIS. Tech. rep., Antiy Labs (2018), <https://www.antiy.net/p/technical-analysis-of-industrial-control-malware-trisis/>
8. Antonioli, D., Tuppenhauer, N.O.: MiniCPS: A toolkit for security research on CPS networks. In: *Proceedings of the First ACM workshop on cyber-physical systems-security and/or privacy*. pp. 91–100 (2015)
9. Basnight, Z., Butts, J., Lopez Jr, J., Dube, T.: Firmware modification attacks on programmable logic controllers. *International Journal of Critical Infrastructure Protection* **6**(2), 76–84 (2013)

10. Brodt, O.: A Brief History of ICS-Tailored Attacks (2023), <https://www.darkreading.com/cyberattacks-data-breaches/brief-history-of-ics-tailored-attacks>, [Accessed: Jan. 14, 2024]
11. Brubaker, N., Lunden, K., Proska, K., Umair, M., Zafra, D.K., Hildebrandt, C., Caldwell, R.: INCONTROLLER: New State-Sponsored Cyber Attack Tools Target Multiple Industrial Control Systems | Mandiant | Google Cloud Blog. <https://cloud.google.com/blog/topics/threat-intelligence/incontroller-state-sponsored-ics-tool> (2022), [Accessed 21-01-2025]
12. Bulazel, A., Yener, B.: A survey on automated dynamic malware analysis evasion and counter-evasion: PC, mobile, and web. In: Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium. pp. 1–21 (2017)
13. Cabaj, K., Plamowski, S., Chaber, P., Ławryńczuk, M., Marusak, P., Nebeluk, R., Wojtulewicz, A., Zarzycki, K.: Cyber4ot dataset: Network traces for cyber-security vulnerability evaluation in industrial control systems. *SoftwareX* **31**, 102196 (2025)
14. Caselli, M., Hadžiosmanović, D., Zambon, E., Kargl, F.: On the feasibility of device fingerprinting in industrial control systems. In: International Workshop on Critical Information Infrastructures Security. pp. 155–166. Springer (2013)
15. Cherepanov, A.: Win32/Industroyer. A New Threat for Industrial Control Systems. https://web-assets.esetstatic.com/wls/2017/06/Win32_Industroyer.pdf (2017), [Accessed 21-01-2025]
16. Chung, K., Kalbarczyk, Z.T., Iyer, R.K.: Availability attacks on computing systems through alteration of environmental control: smart malware approach. In: Proceedings of the 10th ACM/IEEE international conference on cyber-physical systems. pp. 1–12 (2019)
17. Conti, M., Donadel, D., Turrin, F.: A survey on industrial control system testbeds and datasets for security research. *IEEE Communications Surveys & Tutorials* **23**(4), 2248–2294 (2021)
18. Cordeiro, A., Vasilomanolakis, E.: Towards agnostic operational technology (ot) honeypot fingerprinting. In: 9th Network Traffic Measurement and Analysis Conference. IFIP (2025)
19. Cousineau, D., Inoue, H., Marché, C., Mentré, D.: A Methodological Guide for the Validation of Logic Modelling of Ladder Instructions. Ph.D. thesis, Inria (2024)
20. Damodaran, A., Troia, F.D., Visaggio, C.A., Austin, T.H., Stamp, M.: A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques* **13**(1), 1–12 (2017)
21. Dehlaghi-Ghadim, A., Moghadam, M.H., Balador, A., Hansson, H.: Anomaly detection dataset for industrial control systems. *IEEE Access* **11**, 107982–107996 (2023)
22. Donadel, D., Crestanello, G., Morandini, G., Antonioli, D., Conti, M., Merro, M.: Simprocess: High fidelity simulation of noisy ics physical processes. In: Proceedings of the 11th ACM Cyber-Physical System Security Workshop. pp. 6–18 (2025)
23. Dragos, Inc.: PIPEDREAM: CHERNOVITE’s emerging malware targeting industrial control systems. Tech. rep., Dragos, Inc. (Apr 2022), <https://hub.dragos.com/whitepaper/chernovite-pipedream>
24. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)* **44**(2), 1–42 (2008)
25. Ekisa, C., Briain, D.Ó., Kavanagh, Y.: Vicsort-a virtualised ics open-source research testbed. In: 2022 Cyber Research Conference-Ireland (Cyber-RCI). pp. 1–8. IEEE (2022)

26. F-Secure Labs: Havex Hunts For ICS/SCADA Systems. <https://archive.f-secure.com/weblog/archives/00002718.html> (2014), [Accessed 21-01-2025]
27. Falliere, N., Murchu, L.O., Chien, E., et al.: W32. stuxnet dossier. White paper, symantec corp., security response **5**(6), 29 (2011)
28. Faramondi, L., Flammini, F., Guarino, S., Setola, R.: A hardware-in-the-loop water distribution testbed dataset for cyber-physical security testing. *IEEE Access* **9**, 122385–122396 (2021)
29. Formby, D., Durbha, S., Beyah, R.: Out of control: Ransomware for industrial control systems. In: RSA conference. vol. 4, p. 8 (2017)
30. Formby, D., Srinivasan, P., Leonard, A.M., Rogers, J.D., Beyah, R.A.: Who’s in control of your control system? device fingerprinting for cyber-physical systems. In: NDSS (2016)
31. Frazão, I., Abreu, P.H., Cruz, T., Araújo, H., Simões, P.: Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process. In: International Conference on Critical Information Infrastructures Security. pp. 230–235. Springer (2018)
32. Gaggero, G.B., Armellin, A., Portomauro, G., Marchese, M.: Industrial control system-anomaly detection dataset (ics-add) for cyber-physical security monitoring in smart industry environments. *IEEE Access* **12**, 64140–64149 (2024)
33. Garcia, L., Brasser, F., Cintuglu, M.H., Sadeghi, A.R., Mohammed, O.A., Zonouz, S.A.: Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In: NDSS. pp. 1–15 (2017)
34. Govil, N., Agrawal, A., Tippenhauer, N.O.: On ladder logic bombs in industrial control systems. In: Computer Security: ESORICS 2017 International Workshops, CyberICPS 2017 and SECPRE 2017, Oslo, Norway, September 14-15, 2017, Revised Selected Papers 3. pp. 110–126. Springer (2018)
35. Hemsley, K.E., Fisher, E., et al.: History of industrial control system cyber incidents. Tech. rep., Idaho National Lab.(INL), Idaho Falls, ID (United States) (2018)
36. Holdbrook, R., Odeyomi, O., Yi, S., Roy, K.: Network-based intrusion detection for industrial and robotics systems: A comprehensive survey. *Electronics* **13**(22), 4440 (2024)
37. Homan, J., McBride, S., Caldwell, R.: IRONGATE ICS malware: Nothing to see here... masking malicious activity on SCADA systems. Tech. rep., FireEye (Jun 2016), <https://cloud.google.com/blog/topics/threat-intelligence/irongate-ics-malware>
38. Jicha, A., Patton, M., Chen, H.: SCADA honeypots: An in-depth analysis of Conpot. In: 2016 IEEE conference on intelligence and security informatics (ISI). pp. 196–198. IEEE (2016)
39. Johnson, B., Caban, D., Krotofil, M., Scali, D., Brubaker, N., Glyer, C.: Attackers deploy new ICS attack framework “TRITON” and cause operational disruption to critical infrastructure. Tech. rep., Mandiant / FireEye (Dec 2017), <https://cloud.google.com/blog/topics/threat-intelligence/attackers-deploy-new-ics-attack-framework-triton>
40. Johnson, B., Caban, D., Krotofil, M., Scali, D., Brubaker, N., Glyer, C.: Attackers deploy new ICS attack framework “TRITON” and cause operational disruption to critical infrastructure. *Threat Research Blog* **14**, 94 (2017)
41. Kalle, S., Ameen, N., Yoo, H., Ahmed, I.: CLIK on PLCs! Attacking control logic with decompilation and virtual PLC. In: Binary Analysis Research (BAR) Workshop, Network and Distributed System Security Symposium (NDSS). pp. 1–12 (2019)

42. Kapravelos, A., Cova, M., Kruegel, C., Vigna, G.: Escape from monkey island: Evading high-interaction honeyclients. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 124–143. Springer (2011)
43. Keliris, A., Maniatakos, M.: ICSREF: A Framework for Automated Reverse Engineering of Industrial Control Systems Binaries. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019. The Internet Society (2019)
44. Koucham, O., Mocanu, S., Hiet, G., Thiriet, J.M., Majorczyk, F.: Detecting process-aware attacks in sequential control systems. In: Nordic Conference on Secure IT Systems. pp. 20–36. Springer (2016)
45. Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. *IEEE security & privacy* **9**(3), 49–51 (2011)
46. Lucchese, M., Lupia, F., Merro, M., Paci, F., Zannone, N., Furfaro, A.: Honey-ICS: A high-interaction physics-aware honeynet for industrial control systems. In: Proceedings of the 18th International Conference on Availability, Reliability and Security. pp. 1–10 (2023)
47. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Curran Associates, Inc. (2017), <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>
48. Mandiant: IRONGATE ICS Malware: Nothing to See Here...Masking Malicious Activity on SCADA Systems | Google Cloud Blog. <https://cloud.google.com/blog/topics/threat-intelligence/irongate-ics-malware> (2016), [Accessed 21-01-2025]
49. Martinez, C.V., Vogel-Heuser, B.: A host intrusion detection system architecture for embedded industrial devices. *journal of the Franklin Institute* **358**(1), 210–236 (2021)
50. Mathur, A.P., Tippenhauer, N.O.: SWaT: A water treatment testbed for research and training on ICS security. In: 2016 international workshop on cyber-physical systems for smart water networks (CySWater). pp. 31–36. IEEE (2016)
51. Mekdad, Y., Bernieri, G., Conti, M., El Fergougui, A.: The rise of ICS malware: A comparative analysis. In: European Symposium on Research in Computer Security. pp. 496–511. Springer (2021)
52. Miramirkhani, N., Appini, M.P., Nikiforakis, N., Polychronakis, M.: Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 1009–1024. IEEE (2017)
53. Murillo, A., Taormina, R., Tippenhauer, N., Galelli, S.: Co-simulating physical processes and network data for high-fidelity cyber-security experiments. In: Sixth Annual Industrial Control System Security (ICSS) Workshop. pp. 13–20 (2020)
54. Rrushi, J., Farhangi, H., Howey, C., Carmichael, K., Dabell, J.: A quantitative evaluation of the target selection of havex ics malware plugin. In: Industrial control system security (ICSS) workshop. pp. 1–5. ICSS (2015)
55. Ruggia, A., Nisi, D., Dambra, S., Merlo, A., Balzarotti, D., Aonzo, S.: Unmasking the Veiled: A Comprehensive Analysis of Android Evasive Malware. In: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security. pp. 383–398 (2024)
56. Salazar, L., Castro, S.R., Lozano, J., Koneru, K., Zambon, E., Huang, B., Baldick, R., Krotofil, M., Rojas, A., Cardenas, A.A.: A tale of two Industroymers: It was the season of darkness. In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 312–330. IEEE (2024)

57. Salazar, L., López-Morales, E., Lozano, J., Rubio-Medrano, C., Cardenas, A.A.: ICSNet: A Hybrid-Interaction Honeynet for Industrial Control Systems. In: Proceedings of the Sixth Workshop on CPS&IoT Security and Privacy. pp. 68–79 (2024)
58. Sharma, A., Gupta, B.B., Singh, A.K., Saraswat, V.: Orchestration of APT malware evasive manoeuvres employed for eluding anti-virus and sandbox defense. *Computers & Security* **115**, 102627 (2022)
59. Shekari, T., Cardenas, A.A., Beyah, R.: MaDIoT 2.0: Modern High-Wattage IoT botnet attacks and defenses. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 3539–3556 (2022)
60. Shen, C., Liu, C., Tan, H., Wang, Z., Xu, D., Su, X.: Hybrid-augmented device fingerprinting for intrusion detection in industrial control system networks. *IEEE Wireless Communications* **25**(6), 26–31 (2018)
61. Shrivastava, S.: Blackenergy-malware for cyber-physical attacks. *Singapore* **74**, 115 (2016)
62. Soltan, S., Mittal, P., Poor, H.V.: BlackIoT: IoT botnet of high wattage devices can disrupt the power grid. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 15–32 (2018)
63. Sun, Y., Tian, Z., Li, M., Su, S., Du, X., Guizani, M.: Honeypot identification in softwarized industrial cyber-physical systems. *IEEE Transactions on Industrial Informatics* **17**(8), 5542–5551 (2020)
64. Team82: Unpacking the Blackjack Group’s Fuxnet Malware. <https://claroty.com/team82/research/unpacking-the-blackjack-groups-fuxnet-malware> (2014), [Accessed 21-01-2025]
65. Vedere Labs (Forescout) and TU/Eindhoven: Industroyer2 and INCONTROLLER. In-depth Technical Analysis of the Most Recent ICS-specific Malware. <https://www.forescout.com/resources/industroyer2-and-incontroller-report/> (2022), [Accessed 21-01-2025]
66. Williams, T.J.: The purdue enterprise reference architecture. *Computers in industry* **24**(2-3), 141–158 (1994)
67. Zamiri-Gourabi, M.R., Qalaei, A.R., Azad, B.A.: Gas what? I can see your gaspots. studying the fingerprintability of ICS honeypots in the wild. In: Proceedings of the annual industrial control system security (ICSS) workshop. pp. 30–37 (2019)
68. Zhou, X., Cheng, Z., Wang, C., Wang, S., Tao, C., Zhou, Z., Chen, X., Luo, J., Wang, D., Zhou, H.: A dataset collected in real-world industrial control systems for network attack detection. *Scientific Data* (2026)

A Appendix

Table 6 describes the different datasets employed, highlighting the assigned class and the protocol employed, in addition to the source from which the dataset was obtained.

Table 7 lists the features employed in ShadowICS, after variance-thresholding cleanup. For the network column, the entries listed as **global** are used only for high access.

Table 6: Datasets and simulators employed in this paper after removing entries with no physical processes in the network traffic.

Dataset Name	Class	Protocol	Reference
qut-s7	Real	S7Comm	github.com/qut-infosec/2017QUT_S7comm
WDT	Real	MODBUS	[28]
cic-2023-hmi	Simulation	MODBUS	www.unb.ca/cic/datasets/modbus-2023.html
cic-2023-netWide	Simulation	MODBUS	www.unb.ca/cic/datasets/modbus-2023.html
ics-flow	Simulation	MODBUS	[21]
vicsort-icszone	Simulation	MODBUS	[25]
dhalsim	Simulation	CIP	[53]
wadi	Real	CIP	[5]
honeyics	Simulation	MODBUS	[46]
4sics	Simulation	MODBUS	www.netresec.com/?page=PCAP4SICS
ics-simlab	Simulation	MODBUS	github.com/JaxsonBrownie/ICS-SimLab
swat-a8	Real	CIP	[50]
swat-a6	Real	CIP	[50]
det-proc	Real	MODBUS	[44]
ICS-NAD-Schneider	Real	MODBUS	[68]
ICS-NAD-Siemens	Real	S7Comm	[68]
Cyber4OT	Real	MODBUS	[13]
ctf-ics-traffic	Real	S7Comm	github.com/NewBee119/ctf_ics_traffic/
ics-add	Simulation	MODBUS	[32]
minicps	Simulation	CIP	[8]
modbus-tcp-scada	Real	MODBUS	[31]
vicstort-dmz	Simulation	MODBUS	[25]
swat-nefics	Simulation	MODBUS	[57]

Table 7: List of features employed in ShadowICS.

Process Features	Network Features	Timing Features
P-avg-std_normalized	N-ip_df	T-avg_delta_Req2Req
P-std-std_normalized	N-tcp_syn	T-std_delta_Req2Req
P-avg-steady_process	N-tcp_syn_ack	T-avg_delta_Req2Req_tcp
P-std-steady_process	N-tcp_fin	T-std_delta_Req2Req_tcp
P-avg-snr	N-tcp_ack	T-avg_delta_Res2Res
P-std-snr	N-tcp_sack_perm	T-std_delta_Res2Res
P-avg-conf_noise_extraction	N-tcp_timestamp	T-avg_delta_Res2Res_tcp
P-std-conf_noise_extraction	N-tcp_reused_sessions_rate	T-std_delta_Res2Res_tcp
P-avg-variance	N-tcp_max_payload_size	T-avg_delta_Res2Req
P-std-variance	N-packet_rate	T-std_delta_Res2Req
P-avg-permutation_entropy	N-icmp	T-avg_delta_Res2Req_tcp
P-std-permutation_entropy	N-udp	T-std_delta_Res2Req_tcp
P-avg-n_peaks	N-udp_industrial	T-avg_delta_Req2Res
P-std-n_peaks	N-industrial_mac_ratio	T-std_delta_Req2Res
P-avg-kurtosis	N-broadcast_eth_packets_count	T-avg_delta_CLRT
P-std-kurtosis	N-avg_ip_ttl	T-std_delta_CLRT
P-avg-longest_above_mean	N-std_ip_ttl	T-avg_delta_RTT
P-std-longest_above_mean	N-avg_tcp_psh_byte_sent	T-std_delta_RTT
P-avg-longest_below_mean	N-std_tcp_psh_byte_sent	T-avg_delta_RTT_tcp
P-std-longest_below_mean	N-possibly_fake_mac_addresses_count	T-std_delta_RTT_tcp
P-avg-skewness	N-tcp_anomaly_count	T-avg_delta_iRTT
P-std-skewness	N-tcp_keep_alive_count	T-std_delta_iRTT
P-avg-adf_pvalue	N-tcp_advanced_features_count	T-protocol
P-std-adf_pvalue	N-tcp_control_flags_count	
P-avg-max_autocorrelation	N-tcp_name_resolution_count	
P-std-max_autocorrelation	N-tcp_configuration_features_count	
P-avg-noise_avg	N-tcp_link_discovery_count	
P-std-noise_avg	N-unique_tcp_session	
P-avg-noise_std	N-avg_tcp_payload_size	
P-std-noise_std	N-std_tcp_payload_size	
P-avg-noise_median	N-max_tcp_payload_size	
P-std-noise_median	N-mode_tcp_payload_size	
P-avg-noise_root_mean_square	N-avg_tcp_industrial_payload_size	
P-std-noise_root_mean_square	N-std_tcp_industrial_payload_size	
P-avg-noise_binned_entropy	N-max_tcp_industrial_payload_size	
P-std-noise_binned_entropy	N-mode_tcp_industrial_payload_size	
P-avg-noise_number_crossing_m	N-avg_time_relative	
P-std-noise_number_crossing_m	N-std_time_relative	
P-avg-noise_fft_energy_total	N-avg_time_delta	
P-std-noise_fft_energy_total	N-std_time_delta	
P-avg-noise_spectral_centroid	N-global-unique_mac_addresses_count	
P-std-noise_spectral_centroid	N-global-possibly_fake_mac_addresses_count	
P-avg-noise_spectral_entropy	N-global-unique_ttl_over_devices	
P-std-noise_spectral_entropy	N-global-duplicate_ip_warnings	
P-avg-noise_dominant_frequency	N-global-packet_rate	
P-std-noise_dominant_frequency	N-global-oui_ics_rate	
P-avg-noise_count_above_mean	N-global-udp	
P-std-noise_count_above_mean	N-global-discovery_traffic	
P-avg-noise_count_below_mean	N-global-icmp	
P-std-noise_count_below_mean	N-global-dns	
P-avg-CrossCorrMaxOthers		
P-std-CrossCorrMaxOthers		
P-avg-CrossCorrMinOthers		
P-std-CrossCorrMinOthers		
P-avg-CrossCohMaxOthers		
P-std-CrossCohMaxOthers		
P-avg-CrossCohMinOthers		
P-std-CrossCohMinOthers		